

CDU 681.3 : 658.3.053

Ergonomie des logiciels : approche psycho-ergonomique de l'interaction homme-ordinateur

Etude bibliographique

E. Brangier ([1])

Software ergonomics : psycho-ergonomical approach to human computer interaction
A literature review

One of the main reasons why operators are reluctant to the computerization of their work, is the lack of compatibility between the software, the characteristics of the task and the way of thinking of the user.

In this paper, a large number of today's indicators of man-software compatibility, is analysed according to the perceptive, motor, linguistic and cognitive level of man-software interaction.

In this view, a software appears to be « ergonomic » when man-software compatibility is the best possible at the various levels of interaction.

Ergonomics / Software / Computer / Man-machine / Communication

Une des principales raisons expliquant les résistances des opérateurs à l'informatisation de leur poste de travail est le manque de compatibilité entre le logiciel, les caractéristiques de la tâche et les modes de raisonnement de l'utilisateur.

Dans cette note bibliographique, un grand nombre d'indicateurs actuels de la compatibilité homme-logiciel sont analysés en fonction des niveaux d'interaction homme-logiciel (perceptif, moteur, linguistique et au niveau global de l'activité).

Dans cette perspective, un logiciel apparaît « ergonomique » lorsqu'aux divers niveaux de l'interaction, la compatibilité homme-logiciel est la meilleure possible.

Ergonomie / Logiciel / Informatique / Communication homme-machine

1. INTRODUCTION

Aujourd'hui, du site informatique centralisé aux millions de minitels implantés en France, l'informatique laisse son empreinte dans beaucoup de nos comportements. Présent à la fois dans les champs professionnels et domestiques, l'ordinateur n'est plus l'objet rare des années 1970. Mais, cette machine qui ne fait qu'« ordonner » de l'information ne serait rien sans ses logiciels. Dans son acception générale, un logiciel est une partie d'un système informatique, un programme ou une somme de sous-programmes qui déterminent, contrôlent et gèrent les opérations réalisées par l'ordinateur.

([1]) Informatique-CDC, Recherche, développement et techniques avancées, 4 rue Berthollet, BP 16, 94110 Arcueil et Université de Metz, Laboratoire de recherche en sciences humaines et sociales, Ile du Saulcy, 57000 Metz.

L'intérêt des concepteurs et des décideurs pour l'étude des dialogues homme-logiciel s'est renforcé avec l'évolution qu'ont connue les logiciels (Chandon, 1985). Tout d'abord ré-

servés aux spécialistes, ils étaient en quelque sorte des produits « clé en main ». Puis leurs usages se banalisant dans le grand public, ils deviennent des articles de « prêt-à-porter » devant satisfaire des opérateurs très divers dans des applications variées, sous peine d'être sous-employés. Sur le plan commercial, le caractère ergonomique d'un logiciel est un critère de marketing fondé sur la qualité des conditions de travail. Le label « logiciel ergonomique » devient un argument de vente au même titre que celui de « chaise ergonomique ».

Pour les psychologues, médecins du travail et biomécaniciens, s'est posé le problème de l'interface homme-ordinateur, c'est-à-dire de la compatibilité entre les caractéristiques matérielles et logicielles de l'ordinateur et les caractéristiques physiologiques et mentales de l'opérateur humain. Ainsi, les premières études portant sur la charge de travail sur terminal ont cherché, d'une part, à préciser les facteurs de contrainte liés à la tâche, à l'environnement et au poste de travail, et, d'autre part, à évaluer les astreintes : fatigue visuelle, fatigue posturale ou charge psycho-sensorielle (Floru et Cail, 1986). D'autres travaux s'orientent plutôt vers la recherche de la compatibilité du logiciel avec les modes de raisonnement de l'opérateur. Cette dernière approche, sur laquelle est centrée cette revue de la question, constitue la problématique centrale de l'ergonomie des logiciels.

Au niveau des conditions de travail, l'informatisation des postes de travail pose des problèmes que nous allons brièvement analyser, afin de mieux définir la place de l'ergonomie des logiciels au sein des sciences et techniques de l'adaptation des machines à l'homme.

1.1. Considérations générales sur les problèmes ergonomiques liés à l'informatisation

L'introduction de l'informatique dans une entreprise modifie l'organisation du travail et les relations sociales de travail (Balle et Peaucelle, 1972 ; Bird, 1981 ; Crozier, 1980 ; Eksl et Sole, 1979). De plus, l'image de l'informatique est négative pour certains employés qui craignent la perte de leur emploi (Pastré, 1984). Ces transformations peuvent expliquer pour une part le rejet de l'informatisation par certains utilisateurs. Néanmoins, la pertinence de ces interprétations psychosociologiques ne rend pas compte de la totalité de ce rejet. Comme le souligne Senach (1986), cette résistance s'explique également par le manque de compatibilité entre le logiciel, les procédures de travail et les caractéristiques mentales des utilisateurs. Alors que les informaticiens jugent la qualité d'un logiciel à partir de ses performances techniques, les usagers l'apprécient en fonction de sa souplesse d'utilisation, de sa fiabilité, de son homogénéité et de sa compatibilité avec leur mode de raisonnement (Piganiol, 1984a). Dans bien des cas, les nouvelles technologies ne sont pas adaptées à la logique du fonctionnement cognitif des utilisateurs du fait d'un « décalage » entre les objectifs de conception et les objectifs d'utilisation. Par conséquent, elles augmentent la charge mentale de travail de l'opérateur au lieu de la réduire. Aujourd'hui, ce rapport doit s'inverser, autrement dit le logiciel doit être adapté aux caractéristiques de l'opérateur qui l'utilise. Il doit répondre aux exigences des tâches et aux besoins des utilisateurs sous peine d'être inefficace, voire refusé. En conséquence, les informaticiens doivent intégrer l'ergonomie dans la conception des dialogues et donc dans la conception même du logiciel.

1.2. L'ergonomie des logiciels : essai de définition

Pour l'Agence nationale pour l'amélioration des conditions de travail, « l'ergonomie des logiciels a pour objectif de permettre une meilleure adéquation des programmes informatiques aux besoins réels des utilisateurs » (Le Bourgeois et Valentin, 1986). Bisseret (1982) précise et affine cette définition. Pour lui, l'ergonomie des logiciels est « l'adaptation du logiciel à l'usager en tant qu'il traite de l'information. En d'autres termes, l'objectif est d'adapter les comportements de l'ordinateur, c'est-à-dire ses manifestations externes, au fonctionnement cognitif de l'usager dans le sens évidemment d'une assistance la plus efficace et la moins contraignante » possible. La finalité de l'ergonomie des logiciels devient alors « d'apporter aux informaticiens des connaissances sur la manière dont « fonctionne » un sujet donné sur le plan intellectuel dans une classe de problème donnée » (Bisseret, 1985) ; ceci afin de fournir aux opérateurs des informations signifiantes et pertinentes.

D'une manière générale, l'ergonomie des logiciels se définit comme étant la ***discipline étudiant la conception et l'utilisation des interfaces homme-ordinateur, dans le but de permettre la meilleure compatibilité possible entre les opérateurs et les systèmes informatiques.*** Cette compatibilité communicationnelle peut être recherchée aux différents niveaux de l'interaction : perceptivo-moteur, linguistique et au niveau global de l'activité.

Les préoccupations de l'ergonomie des logiciels sont de trois ordres :

a) elle vise à la conception de langages de commande, d'utilisation et/ou de programmation adaptés aux stratégies de l'utilisateur dans sa façon de concevoir, de planifier et/ou de programmer son action, son travail ;

b) elle se centre également sur l'apprentissage de l'utilisation des logiciels ainsi que sur les possibilités d'assistance à l'opérateur dans l'utilisation du logiciel (manuel d'utilisation, aides informatisées incrémentées dans les logiciels...) ;

c) elle élabore des aides informatiques au travail. Il s'agit des aides au diagnostic, au raisonnement et à la surveillance de processus industriels où la construction même du système d'aide fait intervenir une modélisation du comportement et/ou du raisonnement humain (c'est la cas notamment des systèmes experts).

Dans cette note bibliographique, nous aborderons seulement les deux premiers points : la conception de logiciels compatibles avec les modes de fonctionnement des utilisateurs et les aides aux opérateurs. Le dernier point – élaboration des systèmes d'aide informatique à la surveillance de processus complexes – ne sera pas traité ici. On pourra consulter à ce propos Bisseret (1984), Hoc (1985), Brangier (1987).

1.3. La recherche de la compatibilité homme-logiciel

Pour Streitz (1987), l'ergonomie des logiciels dépend en première instance de la compatibilité entre les deux entités du couple utilisateur-logiciel. Cet auteur conceptualise la notion de compatibilité de la façon suivante :

– soit L(f) la réalisation par le logiciel d'une fonctionnalité f (telle que par exemple « copier », « déplacer », « enregistrer », « dessiner »...) ;

– soit U(f) le modèle mental propre à l'utilisateur de la fonctionnalité f ;

– soit U(L(f)) la représentation cognitive qu'a l'utilisateur de la réalisation de la fonctionnalité f par le logiciel.

Plus un système sera compatible, moins la notion de désaccord entre les modèles de connaissances en jeu sera marquée. Dit autrement, moins il y aura de divergence entre L(f), U(f) et U(L(f)), plus le système sera orienté utilisateur. La thèse de Streitz débouche sur deux conséquences. Premièrement, la meilleure compatibilité possible entre un utilisateur et un logiciel passe par la recherche de la meilleure compatibilité possible entre le modèle du concepteur C(f) et celui de l'utilisateur U(f). En d'autres termes, les informaticiens doivent intégrer les modes de fonctionnement des utilisateurs dans la conception même du logiciel. Deuxièmement, dans la quête de la compatibilité, le concepteur est confronté à trois questions (Rialle, 1988) :

a) quelles sont les caractéristiques de l'interface qui permettront à l'opérateur de se construire une représentation qui soit adaptée ?

b) comment est-il possible de faciliter l'apprentissage du fonctionnement d'un logiciel à partir d'un mode de dialogue adapté aux novices vers un mode adapté également aux expérimentés ?

c) les informaticiens doivent-ils prévoir, pour un utilisateur expérimenté, plusieurs types d'interactions qui différeraient par leur compatibilité de diverses situations ou tâches et, si oui, dans quels sens et pour quels effets ?

En s'inspirant de la problématique dessinée par Streitz, nous différencierons trois niveaux de compatibilité qui sont liés aux niveaux d'interactions homme-logiciel.

Premièrement, nous distinguerons les problèmes liés à l'interaction matérielle. Autrement dit, nous analyserons si les dispositifs d'entrée/sortie d'informations sont compatibles avec les caractéristiques de l'opérateur et avec la nature de sa tâche. Par exemple, pour entrer des informations graphiques ou picturales dans un ordinateur, le clavier est quasiment inutilisable. Pour gagner en précision dans le tracé, pour diminuer les erreurs, pour travailler plus vite, l'opérateur doit pouvoir utiliser un dispositif d'entrée compatible avec les exigences motrices de la réalisation de sa tâche (crayon optique, souris ou tablette graphique). De même, au niveau des périphériques de sorties d'informations (écran, imprimante, table traçante...), le dispositif informatique doit tenir compte des caractéristiques perceptives de la réalisation de la tâche par l'opérateur. Dans une tâche de traitement de textes, par exemple, de par leur taille et leur couleur, les caractères doivent être facilement lisibles par l'usager. Plus trivialement : écrire avec des caractères verts sur un écran dont le fond est bleu n'est absolument pas compatible avec une tâche de bureautique, mais cela pourrait être utile et nécessaire à un opérateur travaillant sur un logiciel de conception d'enseignes lumineuses destinées à des devantures de magasins. Nous appellerons ce premier niveau d'interaction celui de la *compatibilité perceptivo-motrice* entre l'homme et le logiciel.

Deuxièmement, les signes linguistiques présents dans un logiciel (mots, phrases, abréviations, codes...) sont parfois arbitraires et n'ont bien souvent de significations que pour l'ordinateur. Dans une entreprise de distribution de biens de consommation courante nous avions constaté que dans un logiciel de gestion de stock, les produits étaient encodés par les opérateurs sous des formes numériques complexes. Par exemple pour encoder une certaine marque de biscottes les utilisateurs devaient taper : « 120020 ». Dans ce cas, l'idéogramme du code ne recouvre pas directement sa signification. C'est le logiciel qui donne au code son sens. Ce type de codification arbitraire est difficilement mémorisable par les opérateurs et représente un facteur augmentant les erreurs d'encodage et la durée de la saisie. De plus, psychosociologiquement, l'opérateur a le sentiment d'être éloigné de l'objet de son travail. S'ajoutant à cela, la saisie devait se faire selon une syntaxe précise imposée par le logiciel. A ce second niveau, la psycho-ergonomie des logiciels traite des problèmes de ***compatibilité linguistique*** entre l'opérateur et le logiciel.

Troisièmement, il s'agit d'analyser comment la structure d'un logiciel est adaptée et adaptable aux modes de raisonnement de l'opérateur mis en jeu dans la réalisation de sa tâche. A ce niveau, l'ergonomie des logiciels se centrera, par exemple, sur la manière dont un opérateur apprend à se servir d'un logiciel. Ce troisième niveau de l'interaction homme-logiciel est celui de la *compatibilité au niveau global de l'activité*.

Ce sont ces trois points que nous allons à présent détailler. Pour des raisons de clarté d'exposé, ces trois niveaux seront distingués. En réalité, ils sont interdépendants.

2. LA COMPATIBILITE HOMME-ORDINATEUR AU NIVEAU PERCEPTIVO-MOTEUR

Il représente le niveau de l'interaction matérielle entre l'opérateur et l'ordinateur. Les problèmes posés à ce niveau de la compatibilité homme-logiciel sont relatifs à l'adéquation des dispositifs d'entrée et de sortie d'information aux caractéristiques motrices et perceptives des opérateurs réalisant leur tâche. Ainsi, cet échelon peut aisément être clivé en deux : d'un côté les problèmes sensori-moteurs liés aux dispositifs d'entrée d'informations, d'un autre côté les problèmes de perception liés aux dispositifs de sortie de l'information.

2.1. L'ergonomie de quelques dispositifs d'entrée d'informations

Actuellement, il existe plusieurs moyens de communiquer avec un logiciel. Le clavier fut bien sûr le premier mode d'entrée d'informations, mais il s'est très vite avéré inadapté aux tâches de dessin graphique ou de poursuite. A partir de cette constatation, d'autres dispositifs ont été conçus : la souris, le crayon optique, la tablette graphique, le joystick ou levier de commande, la boule roulante ou encore, avec les progrès de l'intelligence artificielle, la reconnaissance automatique de la parole. Tous ces dispositifs ont leurs avantages et leurs inconvénients (Janet, 1982). Ils sont jugés selon des critères de maniabilité, de performance et de rapidité dans l'exécution d'une tâche. La qualité ergonomique des dispositifs dépend fortement des caractéristiques de la tâche. Trois types de tâches ont été étudiées expérimentalement : les tâches de désignation (c'est-à-dire désigner une marque sur l'écran avec

un dispositif d'entrée), de poursuite (c'est-à-dire poursuivre un indice avec un dispositif d'entrée) et de dessin. Le tableau I présente un résumé des résultats des études ergonomiques réalisées sur quelques dispositifs fréquents d'entrée d'informations (d'après Janet, 1982 et Valentin et Lucongsang, 1987).

L'évaluation ergonomique de ces divers dispositifs d'entrée d'informations est primordiale pour que le choix de l'utilisateur et du concepteur concernant ces modes de communication ne soit pas seulement guidé par le souci d'une large diffusion. Ce choix doit aussi et surtout maximiser la compatibilité entre la tâche et ses exigences gestuelles. Les dispositifs d'entrée d'informations doivent être cohérents avec la nature de la tâche.

Le critère ergonomique retenu, au niveau de l'interaction matérielle, est celui de la facilité d'utilisation. Elle s'exprime par la cohérence entre les propriétés spatio-temporelles du geste et la structure du langage de commande. Aussi, Denœud et coll. (1986) déconseillent l'association d'un même geste à plusieurs fonctions différentes.

TABLEAU I

Dispositifs	Désignation	Poursuite	Dessin
Clavier	Le plus lent des dispositifs. Cependant il est plus rapide lorsque les données sont courtes	Impossible	Impossible
Crayon optique	Plus rapide et plus sûr que le clavier	Implique de la fatigue gestuelle	Dispositif « naturel » utilisé en conception assistée par ordinateur
Souris	Rapide et fiable	Dispositif le plus rapide et le plus sûr, exige un léger apprentissage	Risque d'erreur à cause des rotations ou perte de contact avec le plan de travail
Touches curseurs	Rapide mais erreurs plus fréquentes qu'avec la souris	Moins rapide et moins sûr que le joystick	Impossible
Joystick	Moins rapide que la souris	Degré de sensibilité et temps de latence élevé	Aussi précis que le crayon optique
Écran sensitif	Idem crayon optique	Difficile	Impossible

2.2. La présentation de l'information sur l'écran

Au niveau perceptif, l'ergonomie des logiciels traite principalement de l'adéquation entre les caractéristiques perceptives de l'homme et la présentation des données. Ce type d'étude permet la définition de principes de présentation des données compatibles avec le système perceptif et cognitif de l'opérateur et avec la nature de sa tâche.

Dans cette perspective, les informations sont étudiées du point de vue de leurs organisations (vitesse de présentation, densité, format d'affichage...) et de leurs codages physiques (couleur, surbrillance, types de caractères...). Nous ne nous étendrons guère sur ces problèmes de perception car ils ont été pris en compte depuis longtemps et sont en partie résolus. Aussi nous renvoyons le lecteur à l'article de Cail (1986).

La solution idéale, permettant une adaptabilité dynamique et non une adaptation statique, consiste à laisser à l'utilisateur la possibilité d'accéder à la mémoire de la machine. L'accès direct à cette mémoire permet à l'opérateur de sélectionner lui-même sur un écran polychrome les couleurs de caractères et de fond. De la même manière, la taille des caractères peut être changée. L'utilisateur peut donc aménager les codages physiques de l'information selon les caractéristiques de sa tâche. Les données physiques visualisées sur l'écran sont donc adaptables par l'opérateur selon ses caractéristiques propres et celles de sa tâche.

Cependant, l'ensemble des problèmes posés par la présentation des informations n'est pas entièrement résolu par l'adaptabilité technique. En effet, les concepteurs doivent prévoir la segmentation optimale de l'écran en diverses zones afin de permettre la meilleure localisation spatiale des informations. De même, la densité d'informations doit être constante d'un écran à l'autre.

Le multifenêtrage (c'est-à-dire lorsque l'utilisateur dispose de plusieurs surfaces, sur le même écran, ayant des contenus et des contenants différents pour effectuer son travail) renvoie directement à la psychologie des activités multiples (Miyata et coll., 1986). C'est un support de dialogue qui rend possible la présence et l'activation à l'écran de plusieurs tâches différentes ou complémentaires. Le fenêtrage permet donc la mémorisation des activités en cours de traitement par leur visualisation sur l'écran. Au niveau de l'organisation des fenêtres, deux orientations sont possibles. Premièrement, le « positionnement libre » qui repose sur le principe des dossiers sur le bureau ou des étiquettes autocollantes. Dans ce cas, l'utilisateur risque de perdre de l'espace s'il y a un recouvrement possible des fenêtres et si l'écran est petit. Néanmoins, le positionnement libre semble très bien adapté aux développeurs qui élaborent des programmes informatiques. Deuxièmement, « le positionnement figé » est fondé sur l'idée du tableau de bord. Il semble améliorer la gestion des fenêtres en facilitant la spatialisation des informations à l'écran. De plus, le positionnement figé réduit les pertes d'espace qui sont liées à la superposition des fenêtres. Alors que choisir ? Encore une fois, l'analyse du travail des opérateurs permettra de tester et de valider le plus adapté des deux modes.

C'est à propos des problèmes de perception de l'information à partir des listings ou des schémas imprimés par l'ordinateur qu'ont été développés des systèmes appelés WYSIWYG (What You See Is What You Get, ce que vous voyez à l'écran est ce que vous imprimez sur le papier). Dans le cas où les systèmes informatiques ne sont pas WYSIWYG, ils exigent de la part des opérateurs des efforts d'abstraction. La perception des états finaux de la tâche doit pouvoir être anticipée à partir de la visualisation de leurs états initiaux sur l'écran, sans quoi l'opérateur est obligé de se représenter ce que sera le résultat imprimé de son travail. Cet écart entre les états imprimés et informatisés peut, bien évidemment, être un facteur d'erreur lorsque, par exemple, l'opérateur définit la mise en page de son édition avec les commandes d'impression.

En conclusion, en matière de présentation de l'information, l'ergonomie des logiciels vise l'adaptation de cette présentation l'exécution de la tâche. Cependant, rappelons qu'actuellement la question fondamentale de l'ergonomie des logiciels n'est plus l'écran et son effet sur l'homme, mais surtout l'utilisation des informations visualisées sur l'écran pour la réalisation d'une tâche.

3. LA COMPATIBILITE HOMME-LOGICIEL AU NIVEAU LINGUISTIQUE

Lors de l'utilisation d'un logiciel, l'opérateur doit à la fois organiser son travail et choisir les traitements à effectuer. Ceci l'oblige à définir un mode opératoire compatible avec les contraintes du dispositif (Michard, 1985). L'opérateur doit également planifier son action et s'informer sur la sémantique et la syntaxe du dispositif. Pour ce faire, il utilise des langages d'interaction lui permettant le démarrage, l'utilisation, la modification d'un logiciel ainsi que la saisie, le traitement, la sauvegarde, le transfert (vers d'autres systèmes informatiques) et l'impression des données. A l'inverse, les langages de programmation (par exemple basic, cobol, lisp, fortran, pascal...) lui sont généralement opaques. Ces derniers ne tolèrent pas les ambiguïtés et sont donc particuliers et artificiels. Nous n'aborderons pas les problèmes ergonomiques qu'ils occasionnent aux informaticiens (à ce propos consulter Hoc, 1982 et Mayer, 1987). Notons que ces langages « ésotériques » évoluent, en s'enrichissant d'une syntaxe et d'une sémantique proche du naturel. C'est, par exemple, le cas des langages de requêtes ou des langages orientés applications.

Prenons l'exemple d'un problème d'ordre linguistique rencontré dans l'utilisation d'un logiciel de gestion du personnel d'une entreprise d'informatique. Dans le fichier informatique, les salariés étaient encodés, entre autres, selon leur qualification et selon leur fonction. La qualification correspondait à la classification du salarié sur une échelle de salaire et de hiérarchie allant du coursier au PDG. La fonction était une donnée plus subjective, elle était le titre ou le statut que le salarié négociait avec le conseiller en recrutement soit lors de l'embauchage, soit lors d'une promotion. Autrement dit, sur la fiche de paie du salarié, le salaire était attribué à travers une grille des qualifications et le statut professionnel lié à la fonction. Les problèmes ergonomiques rencontrés dans un tel dispositif étaient doubles :

– d'une part, il y avait incompréhension entre les personnes des services comptable et recrutement/mobilité. Pour les premiers, la caractéristique dominante d'un employé est sa qualification et, pour les seconds, sa fonction ;

– d'autre part, cette confusion était entretenue par le chevauchement des codifications. Par exemple, à la qualification « analyste » encodée « 13.B », correspondait plusieurs fonctions : « analyste-programmeur » encodée « 13.1 », « analyste » encodée « 13.2 », « analyste expérimenté » encodée « 13.3 » et « analyste-concepteur » encodée « 13.4 ». Notons tout d'abord la difficulté de mémoriser de tels codes chiffrés dont la forme codée n'a pas de lien mnémotechnique avec la signification. Constatons ensuite que le mot « analyste » est codé dans un cas « 13.B » et dans un autre cas « 13.2 ». Le terme « analyste » était à la fois une qualification et une fonction : il n'était pas biunivoque.

Dans cet exemple, l'inadéquation entre les opérateurs et le logiciel se situe au niveau de l'interaction linguistique. Définissons de manière plus précise ce qu'elle recouvre.

La sémantique étudie la relation des signes (mots, phrases, concepts) aux choses et aux états que peuvent avoir ces choses. C'est l'analyse du rapport entre le sens, la vérité et la référence. La syntaxe traite des relations des signes entre eux : rapport des mots avec d'autres mots ou phrases. Ainsi, l'approche syntaxico-sémantique d'un logiciel doit décrire et analyser la fiabilité et l'acceptabilité par l'utilisateur des objets (mots, phrases, codes...) composant le dispositif et les opérations réalisables sur ces derniers. C'est l'étude du recouvrement linguistique entre le « jargon » utilisé dans la tâche informatisée et celui utilisé dans la tâche réelle. La compétence conversationnelle (Theureau et Pinsky, 1984) d'un logiciel est évaluée à ce niveau.

Afin d'améliorer la compatibilité linguistique entre l'homme et le logiciel, la stratégie est d'adapter le langage du logiciel au langage de l'utilisateur. Dans cet objectif, il existe deux approches qui s'opposent :

1) soit développer des langages restreints car (a) la machine ne peut pas « comprendre » le langage naturel, et car (b) le langage naturel n'est pas toujours plus performant qu'un langage restreint bien conçu (Falzon, 1986) ;

2) soit essayer tant bien que mal d'harmoniser le dialogue en langue naturelle entre l'homme et l'ordinateur. Mais cette deuxième approche reste actuellement une impasse.

L'utilisation de signes linguistiques purement arbitraires (non motivés par une signification concrète de l'objet concerné) permet d'observer des conduites quasi-divinatoires mises en jeu par les opérateurs pour décoder ces signes (Piganiol, 1984a).

3.1. Les problèmes de l'interprétation des signes linguistiques

Par signe linguistique nous entendons tout message de nature alphanumérique, émis par l'opérateur ou par la machine, et visualisable dans l'interface utilisateur. Dans un logiciel nous pouvons distinguer deux types de signes linguistiques :

1) les étiquettes nominales : c'est-à-dire les mots (les codes, les commandes et le lexique du dialogue...) et les phrases (messages d'aide ou d'erreur...), c'est le contenu du dialogue ;

2) les modes d'échange entre l'homme et le logiciel : c'est-à-dire les menus, arborescences... qui imposent un cheminement au dialogue, c'est la forme du dialogue.

3.1.1. Les étiquettes nominales

La production des langages de commande par les concepteurs, comme l'interprétation des messages, repose sur une représentation du fonctionnement de la machine. En effet, un utilisateur novice ou occasionnel aura beaucoup de peine à comprendre ce qu'est une « directory » ou un « batch ». Il lui faudra expérimenter le logiciel pour inférer la signification de ces termes à partir de sa pratique. Pynte (1984) fait remarquer la prépondérance de la représentation machinique dans la mesure où « les énoncés sont réduits à leurs seules composantes utiles vis-à-vis de la tâche ». Cet auteur constate aussi le caractère incohérent et incomplet de la représentation que les opérateurs ont de leur logiciel, ce qui renforce le rôle du contexte dans l'interprétation des messages issus de l'ordinateur. Ainsi, le dialogue homme-logiciel est construit à l'inverse de ce que préconise les théories linguistiques associant des significations stables aux énoncés. En effet, dans un dispositif informatique, l'opérateur n'accède pas directement à la signification de l'énoncé. Par exemple, le terme de « batch » ne peut être compris qu'en rapport avec une procédure ou une tâche. Par conséquent, les mots, les phrases ne sont pas immédiatement opérationnels vis-à-vis de la tâche à réaliser. En d'autres termes, la signification n'est pas contenue dans l'énoncé. Pour comprendre le dialogue, l'utilisateur est obligé d'effectuer une inférence supplémentaire basée sur une analyse linguistique de l'énoncé d'une part, sur une analyse perceptive et/ou conceptuelle de la situation d'autre part, ce qui représente une double tâche : manipulation des concepts et extraction de leur signification.

Dans certains cas, l'usager ne connaît pas exactement le sens des mots qui apparaissent à l'écran. Par ailleurs, les réponses du système sont inflexibles, constamment identiques (c'est notamment le cas des messages d'erreur). De ce fait, l'opérateur ne peut pas se constituer une représentation fidèle du fonctionnement du logiciel étant donné qu'il ne connaît pas le sens des mots qu'il manipule et que les réponses de l'ordinateur ne lui fournissent pas des éléments suffisants de compréhension. Ces défauts du dialogue ont pour effet de gêner l'opérateur dans sa reconstruction mentale de l'architecture du système, de son organisation interne... Dans certains cas, il ne reste que deux possibilités à l'utilisateur : soit tenter de découvrir tant bien que mal le fonctionnement du logiciel, soit laisser le système agir. Voilà pourquoi Piganiol (1984a) affirme que « l'ordinateur crée de faux dialogues ».

Pour remédier aux problèmes d'interprétation des étiquettes nominales, le type de langage d'interaction choisi doit être fondé sur les caractéristiques de la tâche et de l'opérateur. De plus, l'utilisateur doit avoir la possibilité d'adapter les significations évoquées aux contraintes de la tâche et de personnaliser son dialogue avec la machine, ce qui implique des analyses préalables de la tâche pour créer, dès la conception, un langage d'interaction compatible avec le domaine d'application du logiciel.

3.1.2. Des modes d'échanges entre l'homme et le logiciel

Sperandio (1983 et 1987a) note que chacun des modes d'échanges d'information (menus, arborescences questions/réponses, dialogues dirigés) entre l'homme et l'ordinateur présentent des avantages et des inconvénients propres selon leurs contextes d'application. L'analyse préalable du travail doit permettre de prévoir le mode d'échange le mieux adapté à la situation. Ils peuvent être évalués en fonction de leur rapidité, fiabilité et acceptabilité par l'opérateur dans l'application donnée.

L'exemple suivant illustre un problème de choix des étiquettes nominales associées aux nœuds et aux arcs des programmes. Dans Appleworks (logiciel intégré de traitement de texte, tableur et gestion de fichiers), le premier menu apparaissant à l'écran propose à l'utilisateur de travailler sur un fichier alors qu'aucun fichier n'a été créé ou chargé. Si l'opérateur sélectionne la commande de « travailler sur un fichier », l'ordinateur lui indique qu'il ne peut le faire étant donné qu'il n'y a pas de fichier en mémoire. Le logiciel propose une commande qui s'avère, dans l'état initial du système, inutilisable. Le menu doit exclure une telle piste qui ne mène à rien. La nomenclature et le contenu du menu doivent tenir compte des transformations qu'ils peuvent engendrer. Le menu doit clairement indiquer la différence entre un état initial et un état final du système. Dans notre exemple, la commande proposant de travailler sur un fichier présuppose que l'opérateur a créé ou chargé un fichier. Ce sous-entendu ne correspond pas à un état initial mais oblige à une transformation des états du logiciel. Un menu doit clairement indiquer l'inégalité entre les états initiaux et les états où il y a transformation des informations du système, car ils n'ont pas les mêmes statuts pour les utilisateurs. Dans le choix d'une commande (au niveau d'un menu par exemple), il faut que les états initiaux et finaux puissent être comparés. Dans le cas où ils sont différents, il ne doit pas y avoir constance de l'énoncé (Pynte, 1984).

Quant aux nombres d'alternatives d'un menu, l'optimal serait de quatre à huit par page (Lee et MacGregor, 1985) afin que le temps de recherche d'une information par l'opérateur soit dans des limites acceptables. Néanmoins, il est probable que le nombre de choix maximal par menu varie en fonction de la maîtrise que l'opérateur a du système. En effet, les opérateurs confirmés préfèrent fréquemment avoir un premier menu très complet plutôt que d'être obligé de passer par une série de sous-menus qui n'apportent aucune information supplémentaire. Dans les cas où les menus appellent de multiples sous-menus, Kerguelen (1983) signale qu'un grand nombre d'enchaînements peut contrecarrer les procédures anticipées par l'utilisateur. En revanche, cette rigidité peut aussi être un facteur de sécurité dans des procédures dangereuses où l'ordinateur veille sur les manipulations de l'opérateur.

3.2. Le problème du choix des langages restreints

3.2.1. L'impossibilité de dialoguer en langage naturel

L'évolution du dialogue homme-machine est partie d'une structure rigide de la machine (« l'ordinato-centrisme ») pour aller de plus en plus vers un dialogue ayant l'apparence du naturel. Ce mouvement, rendu possible grâce à l'évolution technique, s'est fait en introduisant progressivement les caractéristiques du fonctionnement humain dans l'ordinateur. Il reste un (grand) pas à franchir en lui apprenant le langage. Mais, de nombreux problèmes persistent dans la réalisation de cet ambitieux projet, malgré les grands espoirs de l'intelligence artificielle.

Ainsi, Richard (1983) ne pense pas que la possibilité de dialoguer en langage naturel puisse résoudre l'essentiel des problèmes de communication entre l'homme et l'ordinateur.

Comme l'a démontré Scapin (1982), des difficultés subsistent quand les commandes correspondent à des mots familiers évocateurs pour les opérateurs. Ces noms sont pratiquement toujours polysémiques et la carence du contexte ne permet pas aux opérateurs débutants et/ou occasionnels de lever les ambiguïtés et d'établir des relations stables entre les codes et les fonctions.

Winograd (1984) renforce cette opinion : « c'est l'impossibilité de formaliser le sens contextuel des mots qui empêche pour l'instant (et vraisemblablement pour toujours) de concevoir des logiciels capables de comprendre une langue naturelle comme un être humain ». En fait, un paradoxe de l'ergonomie des logiciels est de rechercher un dialogue naturel, alors que la polysémie des mots oblige les informaticiens, les ergonomes et les utilisateurs à concevoir des langages restreints. Par langage restreint, nous entendons un système linguistique dont la structure syntaxico-sémantique est close, figée et non ambiguë. Le problème psycho-ergonomique de ces langages est la tolérance et l'acceptabilité de la restriction par les opérateurs. En définitive, les langages restreints se sont imposés comme support de communication entre l'opérateur et le logiciel car il est impossible de dialoguer avec une machine en langage naturel et aussi parce que les langages naturels ne sont pas toujours les mieux adaptés aux communications verbales.

En étudiant les dialogues entre les contrôleurs aériens et les pilotes d'avions, Falzon (1986) a montré que ces derniers utilisaient spontanément des langages restreints. Ces langages, bien adaptés à cette situation, permettent une compréhension rapide et fiable des ordres donnés par la tour de contrôle ou par l'équipage. La question fondamentale des langues restreintes dans le dialogue homme-machine est de connaître les restrictions qui soient les plus tolérables par l'utilisateur (Falzon, 1986). Comme nous le verrons plus avant, la règle d'or de cette approche est celle de la biunivocité : une signification et une seule associée à un mot, à une commande ou à un code, et réciproquement.

Un autre problème de la compatibilité linguistique est l'utilisation encore trop fréquente par des opérateurs de logiciels écrits en anglais. En effet, la majorité des logiciels le sont et le manque de traduction oblige les utilisateurs à l'apprentissage de mots anglais. Bien entendu, ceci est un frein à l'optimisation du dialogue homme-logiciel. De la même manière, les stéréotypes de lecture d'une culture imposent souvent un mode de décodage des informations présentes à l'écran. Par exemple « 1.000 » sera lu « mille » en France et « un virgule zéro zéro zéro » en Grande-Bretagne. Ces stéréotypes de lectures doivent être respectés afin de faciliter la tâche de l'opérateur.

Signalons qu'une étude sur la sémantique des codes et des langages de commande ne peut être entreprise sans une analyse des situations de travail, afin de définir le vocabulaire employé par les (futurs) usagers lors de la réalisation de leur tâche et de l'intégrer dans le logiciel.

3.2.2. Les commandes : signification et syntaxe

Ce sont les codes et les langages de commande qui permettent à l'opérateur de dialoguer avec l'ordinateur. Tous ont pour finalité le lancement, l'arrêt, la saisie ou la suppression de certains traitements d'information.

Le nombre de commandes doit être réduit. Le langage de commande doit être organisé « en groupes fonctionnellement similaires » et/ou en groupes à « différents niveaux de complexité pour tenir compte des niveaux d'expérience des utilisateurs » (Scapin, 1985). Richard (1983) conseille de ne pas avoir de commandes à effets multiples car elles ne font qu'augmenter la difficulté de déduire les procédures à partir de la connaissance du fonctionnement global du logiciel.

De plus, même si l'opérateur connaît le résultat d'une commande, cette connaissance s'avère, dans bien des cas, inutilisable immédiatement. L'effet d'une commande ne correspond pas toujours au résultat d'une action qui soit équivalente à un sous-but. Par exemple, le déplacement d'une partie d'un texte dans un logiciel de traitement de texte requiert de la part de l'usager plusieurs inférences. Après avoir sélectionné la commande, l'opérateur doit marquer le texte à déplacer, puis savoir s'il veut déplacer le texte dans le fichier sur lequel il travaille ou sur un autre. Il lui faut encore inférer la nécessité de positionner le curseur à l'endroit désiré et enfin ordonner le déplacement. L'utilisateur doit donc acquérir des règles d'interprétation pour établir les correspondances existant entre les états du système et les commandes (Pynte, 1984).

A propos des touches fonctions, elles sont d'un accès facile, direct et immédiat. Pour des raisons de mémorisation, leur nombre doit être restreint. La signification associée à une touche fonction doit être constante et ce quelle que soit la page écran. Par exemple, quel que soit le logiciel, la touche « F1 » sur les IBM PC (et compatibles) correspond pratiquement toujours à la fonction d'aide.

Quant aux règles syntaxiques des commandes, les solutions ne sont pas universelles, elles ne peuvent être définies qu'à partir d'une analyse du travail, du nombre de fonctions à commander et de leurs multiples connexions (Sperandio, 1983 et 1987a).

3.2.3. Les codes et abréviations

Les codes et abréviations ne concernent que les messages en provenance de l'opérateur. Les informations issues de l'ordinateur ne doivent en aucun cas être abrégées, car cela augmenterait sensiblement la charge mnésique de l'utilisateur. En règle générale, il vaut mieux éviter les abréviations arbitraires. Lorsqu'il faut abréger un mot, on choisira, s'ils sont biunivoques, acceptés et facilement mémorisés par les opérateurs, les trois ou quatre premières lettres ou consonnes du mot désignant la commande (Bonpays, 1985). Néanmoins, pour certains mots, la troncature (suppression d'une partie d'un mot) peut être préférable à la contraction. Par exemple, pour abréger « comptabilité nationale » on choisira plutôt « compta nat » que « cmptblt ntnl ».

L'abréviation choisie dépend aussi du nombre de mots : plus le vocabulaire du dialogue homme-logiciel sera étendu, plus les abréviations seront longues et inversement. Pour Sperandio (1983), l'idéal est le code « naturel et spontané, c'est-à-dire s'établissant sans apprentissage et résistant à l'oubli ».

3.2.4. Le lexique ou vocabulaire du dialogue

Le lexique d'un logiciel, c'est-à-dire l'ensemble du vocabulaire présent dans l'interface utilisateur, doit être

constitué des mots fréquemment employés par les usagers. La définition du vocabulaire de la profession concernée implique une analyse du travail et des dialogues qui s'y rapportent. Le sens des mots doit être clair pour les utilisateurs et chaque mot doit, répétons-le, désigner de façon biunivoque un objet, une commande, une procédure... et inversement (Sperandio, 1987a). La possibilité pour le logiciel d'assimiler des synonymes facilite grandement le dialogue. Là encore, des analyses du travail menées auprès des opérateurs avant la conception faciliteront grandement la définition du lexique du logiciel.

3.2.5. Erreurs, messages d'erreur et protections contre les erreurs accidentelles de l'usager

L'opérateur n'est pas toujours fiable, il lui arrive de commettre des erreurs de frappe ou de commande. Un logiciel adapté devrait pouvoir reconnaître les erreurs les plus fréquentes et les corriger en demandant la confirmation de l'opérateur, plutôt que de l'obliger à effectuer une nouvelle entrée (Scapin, 1985).

L'opérateur n'est pas toujours conscient et/ou informé des erreurs qu'il provoque. Il ne connaît pas a priori les caractéristiques techniques de son erreur. Aussi, les messages d'erreur doivent-ils, d'une part, donner une explication la plus complète possible sur la cause des erreurs, afin que l'opérateur comprenne les dysfonctionnements et, d'autre part, générer automatiquement la suite des actions à entreprendre pour retrouver le fonctionnement normal du système.

Quant aux protections contre les erreurs accidentelles de l'utilisateur, Sperandio (1987a) signale que toutes les actions faites par l'usager doivent pouvoir être annulées sans que les informations traitées le soient. Lorsque l'opérateur entreprend des opérations destructrices (quitter un programme, effacer un fichier...), il est conseillé de doubler les messages d'avertissement signalant l'opération et de veiller à la validation de ces derniers par l'opérateur.

Levis et Norman (1986) posent le problème des caractéristiques de la réponse fournie par le logiciel, alors qu'il ne peut pas interpréter les informations données par l'utilisateur. Ils présentent plusieurs solutions dont une des plus remarquables est sans doute le DWIM (Do What I Mean, implanté sur InterLisp-D). Lorsqu'un utilisateur commet une erreur, DWIM est capable de l'identifier, de reconnaître ce que l'opérateur voulait signifier et de la corriger automatiquement.

3.2.6. La manipulation directe

La manipulation directe d'objets (ou d'icônes) ne s'inscrit pas à proprement parler dans la perspective des langages restreints, elle marque une évolution vers des interfaces plus naturelles.

Le pari des manipulations directes d'objets est de représenter des scénarii communs et simples de travail sous une forme imagée et analogue à une situation naturelle. Par exemple, pour ouvrir un fichier, l'opérateur « clique » (c'est-à-dire agit sur le pointeur de la souris) sur le dessin d'une armoire, cette dernière s'ouvre alors et les noms des dossiers apparaissent. Toujours avec la souris, il est possible de sélectionner l'un d'eux. De même, pour supprimer un fichier, il suffit de « cliquer » sur le dossier à effacer, puis de faire glisser son icône sur le dessin de la poubelle (cas du Macintosh).

La manipulation directe présente un certain nombre d'avantages et d'inconvénients. Côté avantages, le dialogue est reconnu par les usagers comme convivial (Sperandio, 1987a). Une fois que l'apprentissage sensorimoteur de la manipulation de la souris est acquis, l'utilisateur a un meilleur contrôle des actions qu'il entreprend. Cette technique utilise une syntaxe implicite, spontanée et naturelle, demandant peu de mémorisation des commandes et évitant l'emploi et l'apprentissage d'un vocabulaire spécifique. De plus, Hutchins et coll. (1986) notent que :

- les actions effectuées sur les objets sont immédiatement visibles ;
- les utilisateurs occasionnels des interfaces de manipulation directe mémorisent facilement et rapidement les concepts opérationnels du logiciel ;
- l'usager commet moins d'erreurs puisqu'il perçoit directement les résultats de ses manipulations ;
- les opérateurs voient immédiatement si leurs actions correspondent aux buts qu'ils avaient anticipés et peuvent ainsi apporter une correction ;
- enfin, la peur de l'erreur manifestée par les opérateurs diminue quand ils utilisent des interfaces de manipulation directe parce que le système est facilement compréhensible d'une part, et parce que les actions entreprises sont facilement reversibles d'autre part.

Mais ce procédé a également des inconvénients : la manipulation directe d'icônes peut s'avérer lourde à employer pour un utilisateur expérimenté qui recherchera la rapidité d'exécution de son travail. En fait, l'utilisation de la manipulation directe met le doigt sur deux problèmes : d'une part, elle montre clairement qu'un dispositif d'entrée, en l'occurrence la souris, est particulièrement performant dans ce type de tâche, mais, d'autre part, une technique adaptée à des utilisateurs novices ne l'est pas forcément pour des opérateurs expérimentés. Cependant, cette inadaptation est actuellement résolue par le choix laissé à tout moment à l'opérateur : soit cliquer avec la souris sur l'icône de la commande, soit taper directement au clavier le code de la commande. Remarquons également deux problèmes que peuvent générer les icônes. Le premier concerne la « typicalité » de l'objet représenté et sa capacité de généralisation à la classe d'objet qu'il recouvre. Le second est celui de la pertinence du « dessin d'un objet concret pour représenter des actions ou des concepts abstraits » (Sperandio, 1987b).

3.2.7. Les dialogues vocaux-auditifs

Malgré les constantes évolutions de la reconnaissance et de la synthèse vocales, beaucoup de progrès restent à accomplir pour améliorer la qualité des voix et élargir le vocabulaire. Le développement de cette technique, encore très peu utilisée, implique la conception d'interfaces intelligentes qui utiliseront un langage »opératif », naturel ou restreint. Dans ce cadre, le rôle de l'ergonome sera de définir « les contraintes et les fonctionnalités relatives aux tâches et aux utilisateurs » (Sperandio, 1987b).

En dépit des imperfections technologiques, les dialogues vocaux-auditifs sont intéressants quand l'opérateur ne peut pas avoir accès à l'information visuelle (malvoyant, voies visuelles surchargées, déplacements fréquents sur un grand espace, action à distance comme par exemple au téléphone, manque de lumière...) et quand le message est court, simple et qu'il demande une réponse instantanée.

4. LA COMPATIBILITE HOMME-LOGICIEL AU NIVEAU GLOBAL DE L'ACTIVITE

A ce niveau, l'ergonomie des logiciels vise à la compatibilité entre les structures du dialogue d'une part, et les opérations mentales mises en jeu par l'usager lors de la réalisation d'une tâche d'autre part. Autrement dit, elle recherche la meilleure compatibilité possible entre deux représentations de l'objet du travail : la représentation machine choisie par le concepteur et la représentation interne de l'opérateur (Bisseret, 1983a ; Falzon, 1982). C'est un des problèmes centraux de l'optimisation de l'interface homme-machine. Cette compatibilité des systèmes de traitement et de représentation de l'information entre l'opérateur et le logiciel est nécessaire car elle facilite la compréhension et l'apprentissage du fonctionnement de l'application. Elle permet à l'opérateur de reconstruire la logique du dispositif à partir de variables (mots, codes, commandes, menus, icônes, schémas, messages...) prélevées dans le logiciel et donc de l'utiliser de façon optimale.

Concernant cette compatibilité des logiciels avec l'activité des utilisateurs, trois points peuvent être distingués dans la littérature :

- la compatibilité est directement fonction des caractéristiques de la tâche ;
- c'est en réalisant une tâche que l'opérateur doit pouvoir comprendre le fonctionnement du logiciel ;
- cette compatibilité peut être améliorée par des dispositifs d'assistance à l'opérateur intégrés au logiciel (aides logicielles).

4.1. La compatibilité homme-logiciel et la nature des tâches

Un problème concernant la nature de la tâche est l'impossibilité de concevoir un modèle général valable pour l'ensemble des tâches. Par exemple, des problèmes spécifiques à quelques tâches informatisées (la saisie des données, le contrôle de processus, le traitement de texte) ont été étudiés (Sperandio, 1987a). Mais un dispositif adéquat pour une tâche ne l'est pas nécessairement pour une autre. Bien connaître les tâches des opérateurs est donc tout à fait essentiel pour concevoir un logiciel utilisable. Bisseret (1982) insiste sur l'importance de l'analyse du système de travail, ce qui implique :

- la connaissance précise des différentes fonctions assurées par les opérateurs et des modes opératoires qu'ils mettent en jeu ;
- la définition préalable du partage des tâches entre l'homme et le logiciel ;
- la prévision, dès la conception, de l'optimisation de l'interaction opérateur-logiciel.

Cette analyse a priori doit être une « description logique et normative des activités de l'opérateur visant à l'éclaircissement de leur travail réel » (Bisseret, 1982). Elle ne doit pas reproduire exactement l'existant mais l'améliorer et en diminuer les défauts. L'analyse du travail doit également préciser les exigences auxquelles le dispositif informatique devra se soumettre. Elle doit encore favoriser le meilleur recouvrement possible entre la tâche réelle et la tâche informatisée. La méconnaissance de la tâche par les concepteurs implique généralement des erreurs de conception. Ainsi, il faut décrire la tâche et le dialogue s'y rapportant, afin de déterminer le vocabulaire de l'interface (Scapin, 1985). L'analyse doit déboucher à la fois sur la participation des usagers aux jeux d'essais (Valentin et Lucongsang, 1987) et sur la conception d'une maquette devant être testée et validée avec les usagers selon les critères connus de fiabilité, d'homogénéité, d'acceptabilité et de rapidité, pour être ensuite aménagée selon les résultats de ces validations. Néanmoins, il ne s'agit pas de faire effectuer les jeux d'essais par les opérateurs, mais de voir s'ils anticipent les différents cas (calcul, sauvegarde, édition, traitements...) que le logiciel doit effectuer.

N'ayant pas de modèle général des caractéristiques de la tâche informatisée, nous poserons l'hypothèse qu'un modèle facilement utilisable pourrait être celui où l'opérateur peut anticiper les procédures, les actions et les buts futurs offerts par le logiciel quelle que soit sa position au niveau d'un nœud de l'organigramme. Sous cet angle, des indicateurs de l'architecture du fonctionnement du logiciel doivent être inscrits en permanence à l'écran, afin que l'opérateur comprenne la logique du modèle de la tâche informatisée.

Une des caractéristiques générale de toutes les tâches informatisées est leur planification temporelle. A ce propos deux problèmes se posent. Premièrement, un certain nombre d'auteurs, dont Schleiffer (in : Floru et Neboit, 1986), insiste sur le fait que la lenteur du système génère des frustrations. Ainsi, les temps de réponses de l'ordinateur doivent être courts et stables. Dans le cas où les calculs effectués par l'ordinateur dépassent cinq secondes, il est conseillé d'afficher à l'écran la durée approximative du traitement. Ceci dans le but de permettre à l'opérateur de planifier son temps de travail. Deuxièmement, dans ce même objectif, une des caractéristiques de l'adaptabilité pour l'opérateur est de pouvoir interrompre à tous moments son application pour effectuer d'autres transactions ou d'autres tâches (Vanneste, 1987).

4.2. La compréhension et l'apprentissage du fonctionnement d'un logiciel

Lorsqu'un opérateur apprend à utiliser un logiciel dans le cadre d'une application, son objectif premier est de trouver une procédure pour réaliser son travail. Son souci majeur est donc de réaliser sa tâche et non de comprendre le fonctionnement du logiciel (Carroll et Rosson, 1987). Ainsi, tout apprentissage d'un dispositif informatique est d'abord une adaptation aux diverses contraintes de fonctionnement du système (Richard, 1983). Il est donc souhaitable que l'apprentissage soit guidé par les tâches et non par la logique de l'ordinateur (Barthet, 1984).

Pour imprimer un tableau à partir d'un tableur, l'utilisateur doit comprendre la logique de la procédure d'impression ; c'est-à-dire qu'il doit connaître l'ensemble des actions à effectuer (mettre l'imprimante en service, vérifier la position du papier, commander l'impression, déterminer sur l'écran quelle partie du tableau doit être imprimée, définir les commandes d'impression...). Accéder à l'élaboration d'une telle procédure à partir de la connaissance générale du logiciel n'est pas immédiate pour l'opérateur. Le logiciel devra lui permettre de générer, plus ou moins rapidement, une procédure compatible avec les règles de fonctionnement du dispositif. Derechef, pour assimiler la manière exacte dont s'utilise un logiciel « il ne suffit pas d'avoir compris les règles de fonctionnement, il faut opérer un travail de déduction important, qui suppose non seulement des capacités de

déduction suffisantes mais la possibilité de maintenir dans la mémoire de travail les diverses connaissances nécessaires à la dérivation » (Richard, 1983). Ainsi, pour favoriser la compréhension du fonctionnement d'un logiciel, l'opérateur doit avoir à sa disposition des règles d'utilisation relatives aux buts qu'il cherche à atteindre. En d'autres termes, dans notre exemple précédent, les règles d'utilisation permettant l'impression doivent être implantées dans le logiciel et guider l'usager dans la réalisation de son but.

De son côté, Robert (1986) part du constat selon lequel l'opérateur utilise le manuel de référence uniquement lorsqu'il est « bloqué » dans sa découverte du logiciel. Cet auteur a donc choisi d'étudier la manière dont les utilisateurs apprennent seuls le fonctionnement de leur dispositif. Ce type de recherche a un double intérêt. Elle prend en compte la démarche d'exploration du dispositif comme démarche naturelle d'apprentissage. De plus, elle amène à concevoir des logiciels dits de « manipulation directe », c'est-à-dire pouvant être utilisés sans avoir recours à des connaissances informatiques préalables. A partir de ses travaux, l'auteur définit plusieurs caractéristiques de base de ces interfaces :

- l'opérateur doit pouvoir comprendre les transformations produites à telle ou telle position de l'organigramme (création d'un fichier, mises à jour, sauvegarde...) ;

- l'opérateur doit pouvoir comprendre où et pourquoi il se trouve à tel ou tel nœud de l'organigramme (il doit, par exemple, comprendre la logique de l'arborescence du logiciel qui l'amène à effectuer un chargement plutôt qu'une sauvegarde) ;

- le système doit avoir des précautions de sauvegarde de tout ce qui est important ;

- le système doit permettre le retour d'action non désirée vers les actions souhaitées ;

- le système doit garder la trace des actions accomplies par l'opérateur ;

- le logiciel doit éliminer les pistes inutiles ;

- l'architecture du système doit être visible ;

- enfin un aménagement spatial différencié des commandes est souhaitable afin d'attirer l'attention sur les spécificités éventuelles des différentes commandes.

Il est bien évident qu'une interface, permettant à l'opérateur de découvrir naturellement un logiciel, facilite la phase d'apprentissage et donc, à terme, améliore la qualité du dialogue, son efficacité et augmente la rentabilité. Remarquons enfin qu'aujourd'hui encore, on ne sait pas comment un opérateur expérimenté apprend à un opérateur novice le fonctionnement d'un logiciel. Des recherches sur ce transfert de formation et d'expérience permettraient de mieux comprendre les phases délicates de l'apprentissage et d'améliorer en conséquence les modules de formation.

4.3. L'ergonomie des dispositifs d'assistance à l'utilisateur

L'assistance aux opérateurs est constituée d'un ensemble de cours ou de documents informatisés ou non, devant faciliter l'acquisition d'une maîtrise cognitive et/ou « opérative » du dispositif. Ces outils doivent permettre, entre autres, une formation puis une assistance individualisée sur le lieu de travail. Nous évoquerons trois types d'assistance : 1) les manuels d'utilisation ou de référence, 2) les aides informatisées implantées directement dans le logiciel (fonction « Help » Aide) et 3) la formation.

4.3.1. Les manuels d'utilisation

Les manuels de présentation des logiciels doivent permettre l'acquisition des langages d'interaction. C'est en cela qu'ils intéressent l'ergonomie des logiciels. Il est fréquent de constater que les manuels de référence sont sous-utilisés par les usagers. On peut se demander pourquoi ?

On constate généralement que les informations fournies sur le fonctionnement d'un système sont insuffisantes pour que l'utilisateur comprenne aisément comment l'enchaînement des commandes produit le résultat obtenu. L'utilisateur comprendra donc difficilement qu'une suite de commandes soit plus optimale qu'une autre dans un objectif donné. Carroll et Rosson (1987) font remarquer que les manuels sont souvent conçus pour des utilisateurs présupposés déjà expérimentés. De plus, les travaux pratiques associés aux notices d'utilisation sont souvent « trop fragmentaires pour que l'utilisateur ait une idée suffisamment complète des diverses utilisations possibles de l'appareil » (Richard, 1983). A partir d'un exemple anecdotique, l'utilisateur doit inférer la règle générale concrète qui lui permette de transposer une procédure sur plusieurs situations différentes.

Une difficulté majeure rencontrée dans la compréhension des notices d'utilisation consiste en la présentation simultanée de deux modes de description du dispositif. En effet, les manuels centrent leurs explications à la fois sur le fonctionnement du logiciel et sur les résultats correspondant à des objectifs d'actions, qui peuvent être envisagés comme des sous-buts des tâches effectuables mais non spécifiques. En d'autres termes, l'utilisateur trouve dans ces manuels d'une part « des explications qui tiennent au fonctionnement mais qui sont insuffisantes pour fournir un modèle de fonctionnement en termes d'automate et d'autre part des exemples d'utilisation extrêmement spécifiques, qui sont supposés faciliter la compréhension mais qui en réalité se réfèrent à un point de vue complètement différent de celui de la logique du fonctionnement » (Richard, 1983). Les notices d'utilisation devraient parfaitement distinguer la description de l'automate, des objectifs d'actions réalisables.

Senach (1987) ajoute une interprétation supplémentaire concernant les problèmes que rencontrent les lecteurs des notices explicatives. Pour cet auteur, les manuels d'utilisation réduisent, cloisonnent la tâche dans un contenu trop restrictif. Or, cette dernière n'est pas réductible au contenu d'une notice. Une telle réduction revient à dire que le manuel contient toutes les informations nécessaires et suffisantes à la réalisation de la tâche. Mais « un manuel d'utilisation n'est pas un manuel d'apprentissage : l'activité conduit à des acquisitions spécifiques dont le manuel ne rend pas compte ».

Ces observations expliquent pourquoi les opérateurs ne lisent pas ou peu les manuels d'utilisation et découvrent « sur le tas » leurs logiciels. Elles permettent également d'indiquer les principales règles de la rédaction d'un manuel d'utilisation compatible avec l'apprentissage du logiciel.

4.3.2. Les aides informatisées implantées dans les logiciels

Les aides informatisées implantées dans les logiciels apportent à l'utilisateur un soutien pédagogique à la compréhension du fonctionnement de l'application. Leur but est donc de

faciliter l'apprentissage et l'utilisation. On en trouve de deux types : les premières sont des disquettes d'autoformation sur un logiciel ; les secondes sont des sous-parties constitutives du logiciel.

Il ne semble pas exister de recherche ergonomique sur l'efficacité pédagogique des disquettes d'autoformation sur un logiciel. Néanmoins, ces logiciels servant à apprendre d'autres logiciels apparaissent plus, en l'état actuel, comme des démonstrations des possibilités et des performances d'un logiciel que comme des aides pédagogiques destinées aux usagers. Certes, ces logiciels d'autoformation représentent un espoir fondé pour l'ergonomie des logiciels bien que l'on ne connaisse pas leur véritable capacité à faire apprendre une procédure, une commande ou un code...

Pour ce qui est des dispositifs d'aide implantés directement dans un logiciel, Williges (1986) met en évidence la nette amélioration des performances de l'opérateur lorsque le dispositif informatique est accompagné d'un système d'aide. En effet, le temps nécessaire à la réalisation d'une tâche et le nombre d'erreurs par tâche sont significativement plus élevés lorsqu'aucune aide n'est implantée dans le logiciel. Mais cette amélioration d'ensemble masque de profondes disparités entre les systèmes d'aide. Ainsi, les performances de l'opérateur augmentent lorsqu'il est le demandeur de l'aide et diminuent si l'ordinateur propose son assistance. De plus, il semble nécessaire de créer des systèmes d'aide différents selon le niveau d'expérience des opérateurs : des systèmes simples pour les novices, des systèmes très détaillés et complets pour les experts.

A l'heure actuelle, il existe deux types d'implantation de l'assistance à l'opérateur : soit l'aide est concentrée à un nœud et un seul du programme, soit l'aide est « en ligne », c'est-à-dire qu'à chaque étape du déroulement de l'organigramme du logiciel, l'opérateur peut accéder à des explications concernant le niveau où il se trouve (comme Dbase III par exemple). Pour Michard (1985), les aides en ligne sont préférables aux aides groupées car elles facilitent la définition d'un mode opératoire adéquat, non seulement en rappelant la syntaxe des commandes, mais aussi en définissant les enchaînements à suivre pour traiter les différentes informations.

Michard (1985) et Michard et coll. (1988) présentent le système d'aide logiciel « idéal ». Il doit avoir :

– un « analyseur » de langage naturel, pouvant comprendre les questions des utilisateurs ;

– un « générateur » de plan, qui serait approximativement un mini système expert, capable à partir de la connaissance de l'objectif de l'opérateur, de reconstruire l'arbre des transformations à mettre en œuvre ;

– un « éditeur » de plan qui, à partir du plan généré, va construire l'explication conviviale. Les systèmes d'aide « Planex » et « Planex II » ont été conçus à partir de ces principes (Michard, 1985 ; Senach, 1987).

Ainsi, ces aides logicielles favorisent une compréhension de la logique du programme surtout pour les utilisateurs occasionnels.

4.3.3. La formation des utilisateurs sur les logiciels

L'informatisation d'un poste de travail induit bien souvent l'envoi des employés en formation. Il est bien difficile de définir de façon générale le programme que devrait proposer une cession de formation sur des logiciels, tant elle dépend de la nature du travail réel. Néanmoins, Schmalhofer (in : Floru et Neboit, 1986) estime qu'une formation sur un logiciel doit insister sur trois sources d'information :

– la description analytique du dispositif (écran, unité de disquette, ordinateur, imprimante, logiciel...) ;

– des séquences d'exploration interactives (présentation et compréhension de l'organigramme du logiciel) ;

– l'acquisition d'un schéma cognitif.

Bien évidemment, la formation doit être effectuée préalablement à l'implantation d'un nouveau logiciel afin de démystifier la nouvelle technologie et de préparer les utilisateurs à leurs nouvelles qualifications.

Signalons enfin qu'à ce jour, peu de travaux cherchent à évaluer les formations dispensées en informatique ; la mesure de la satisfaction en fin de stage est trop souvent utilisée comme un indicateur de l'efficacité de la formation. Il faudrait aussi rendre compte de l'efficacité du formé sur le terrain. Cette dernière évaluation est bien évidemment difficile à réaliser. Pour une analyse plus approfondie des problèmes de la formation à l'informatique dans le cadre de la bureautique, nous renvoyons le lecteur intéressé à Morel (1987).

5. CONCLUSION

L'utilité réelle d'un logiciel et sa facilité d'utilisation sont d'autant plus grandes qu'il est conçu pour être compatible avec les modes de fonctionnement des utilisateurs sur le plan physiologique (perception, compatibilité des commandes, lisibilité...), sur le plan des tâches (lecture, saisie, dialogue...), sur le plan du langage (codes, lexique...), sur le plan cognitif (modélisation des raisonnements) et psychique (projets personnels...). Comme le soulignent Landauer (1987) et Hammond et coll. (1987), certaines recommandations ergonomiques peuvent être issues des travaux menés en physiologie et en psychologie cognitive mais, si elles sont utiles, elles demeurent insuffisantes et « seule l'analyse du travail, spécificité de l'ergonomie, permet d'approcher la connaissance réelle du travail » (Christol, 1987).

Si le rôle de l'ergonome des logiciels est, et a été principalement, celui d'un conseiller en conception des interfaces utilisateurs, il lui faut maintenant aborder les véritables problèmes qui conditionnent l'efficacité d'un système informatique et qui ne sont pas simplement liés à l'interface mais qui concernent des aspects internes du logiciel (Salembier et Claes, 1987). Ainsi à présent, la formalisation et la modélisation des activités des opérateurs humains devient un champ important d'investigation pour l'ergonomie cognitive et pour la psychologie du travail, notamment lors de la conception des systèmes experts.

Les efforts actuels en ergonomie des logiciels ont surtout consisté à figer une procédure de travail qui paraissait la meilleure. Il semble de plus en plus que les systèmes doivent être adaptables. Encore trop peu abordée aujourd'hui, l'adaptabilité dynamique des logiciels aux caractéristiques des utilisateurs et des tâches connaîtra vraisemblablement un essor avec le développement des interfaces intelligentes et donc de l'intelligence artificielle. A cet égard, notons que les progrès récents de l'intelligence artificielle présagent de grands espoirs : la reconnaissance des voix et des formes, la compréhension des langages naturels, les systèmes experts... sont autant de questions posées à l'ergonomie cognitive des logiciels et auxquelles elle doit répondre pour optimiser l'interaction homme-ordinateur.

Je tiens personnellement à remercier Michel Neboit et Hubert Guillermain pour m'avoir accueilli dans le service Ergonomie, psychologie, sociologie de l'INRS, au cours d'un stage de DEA de psychologie du travail, et pour m'avoir confié la rédaction de cette note bibliographique. Je les remercie également pour leurs conseils et relectures qui ont été sans faille.

Bibliographie

- ALENGRY P. – Evaluation d'un dispositif d'assistance à l'opérateur dans le secteur industriel : ergonomie du dialogue, processus de traitement de l'information, organisation socio-technique. Le Chesnay, INRIA, 1985, 104 p.
- BALLE C., PEAUCELLE J.L. – Le pouvoir informatique dans l'entreprise. Paris, Les éditions d'organisation, 1972, 190 p.
- BARTHET M.F. – Conception de logiciel d'interface homme-machine. Colloque L'ordinateur, l'homme et l'organisation, Nivelles, 5-7 décembre 1984, 9 p.
- BARTHET M.F., PINSKY L. – Les principales techniques de recueil de données en ergonomie des logiciels. In : ALZERA C., CHRISTOL J., FALZON P., MAZOYER B., PINSKY L., SALEMBIER P. (éds) – Ergonomie des logiciels : un atout pour la conception des systèmes informatiques. Paris, La documentation française, collection Les cahiers « Technologie, emploi, travail », 1987, pp. 27-37.
- BARTHET M.F., SEBILLOTTE S. – Les principales techniques de recueil de données en ergonomie des logiciels. In : ALZERA C., CHRISTOL J., FALZON P., MAZOYER B., PINSKY L., SALEMBIER P. (éds) – Ergonomie des logiciels : un atout pour la conception des systèmes informatiques. Paris, La documentation française, collection Les cahiers « Technologie, emploi, travail », 1987, pp. 41-46.
- BIRD E. – Impact des progrès en télécommunications sur les méthodes futures de travail. *Journal des Télécommunications,* 1981, 48, pp. 77-87.
- BISSERET A. – Psychologie ergonomique pour les activités assistées par ordinateur : position des problèmes et exemples de recherche. In : Nouvelles technologies et condition de travail. Liège, Association des médecins, ingénieurs et psychologues du travail, 1980, pp. 3-13.
- BISSERET A. – Pour une psychologie ergonomique des systèmes documentaires. In : Cours de la commission des communautés européennes, Informatique et information scientifique et technique, INRIA, Cap d'Agde, 13-24 septembre 1982, 11 p.
- BISSERET A. – L'ergonomie des interactions homme-ordinateur. Essor d'une psychologie ergonomique pour l'informatique. *Le Travail Humain,* 1983a, 46, 2, pp. 195-198.
- BISSERET A. – Psychology for man computer cooperation in knowledge processing. In : MASON R.E.A. (éd.) – Information Processing. IFIP, 1983b, pp. 113-120.
- BISSERET A. – L'assistance à la résolution de problème dans la supervision de processus. *Intellectica,* 1984, 1, 1, pp. 1-20.
- BISSERET A. – Les hommes et l'informatique. *Le Monde Informatique,* 28 octobre 1985, pp. 43-46.
- BISSERET A., BOUTIN P., MICHARD A. – Eléments introductifs à l'ergonomie des systèmes homme-machine. *Informatique et Sciences Humaines,* 1979, 44, pp. 13-34.
- BONPAYS B. – La saisie interactive, méthode et outil d'une analyse ergonomique. Paris, CNAM, mémoire de DEA, 1985, 41 p.
- BOIS J.P. – Bureautique : la révolution en marche. *Revue Française de Gestion,* septembre-décembre 1985, pp. 177-184.
- BRANGIER E. – Contribution à l'analyse de l'implantation d'un système expert dans une entreprise. Metz, Laboratoire de recherche en sciences humaines et sociales, rapport interne, 1987, 140 p.
- BRANGIER E., MAIRE P., ULRICH O. – Approche psychosociologique de l'informatisation d'une entreprise. Nancy, Laboratoire de psychologie sociale, Groupe de recherche sur les communications, 1985, 315 p.
- CAIL F. – Présentation de l'information sur écran de visualisation. *Cahiers de Notes Documentaires,* 1986, 123, pp. 193-200, ND 1580.
- CARROLL J., ROSSON M. – Paradoxe of the active user. In : CARROLL J. (éd.) – Interfacing thought : cognitive aspect of human computer interaction. Cambridge, The MIT Press, 1987, pp. 80-110.
- CHANDON J.L. – Logiciels : l'obsolescence permanente. *Revue Française de Gestion,* septembre-décembre 1985, pp. 168-176.
- CHRISTOL J. – Les logiciels, un travail pour l'ergonome. In : ALZERA C., CHRISTOL J., FALZON P., MAZOYER B., PINSKY L., SALEMBIER P. (éds) – Ergonomie des logiciels : un atout pour la conception des systèmes informatiques. Paris, La documentation française, collection Les cahiers « Technologie, emploi, travail », 1987, pp. 11-14.
- CROZIER M. – Y a-t-il une stratégie d'introduction de l'informatique ? Paris, La documentation française, Actes du colloque international Informatique et société, 1980, vol. 1.
- DENOEUD B., PAVARD B., VLADIS A., CLEMENT P., FROMONT J., LEJEUNE A., MAUDET J.F. – Command language syntaxe and gestual syntaxe : two constraints in designing usability and learnability. International scientific conference : Work with display units, Stockholm, 12-15 mai 1986.
- ESKI R., SOLE A. – Stratégie d'automatisation, organisation du travail et relations sociales dans les grandes entreprises du tertiaire. *Le Travail Humain,* 1979, 42, 2, pp. 313-324.
- FALZON P. – Structure des dispositifs de présentation de l'information et compatibilité avec les représentations mentales. Congrès de la Société d'ergonomie de langue française, Paris, 6 octobre 1982, texte dacyl., 7 p.
- FALZON P. – Langage naturel et langage restreint. *Le Monde Informatique,* 25 novembre 1985, pp. 36-38.
- FALZON P. – Langages opératifs et compréhension opérative. Paris V, thèse de 3e cycle, 1986, 203 p.
- FLORU R., CAIL F. – Data entry task on VDU : underload or overload. International scientific conference : Work with display units, Stockholm, 12-15 mai 1986.
- FLORU R., NEBOIT M. – Conférence internationale sur le travail sur écran de visualisation. *Cahiers de Notes Documentaires,* 1986, 124, pp. 409-415.
- GREENSTEIN J.S., SUI-TONG L. – An experimental study of dialogue-based communication for dynamic human-computer task allocation. *Journal of Man-Machine Studies,* 1985, 23, pp. 605-621.
- HAMMOND N., GARDINER M., CHRISTIE B., MARSHALL C. - The role of cognitive psychology in user-interface design. In : GARDINER M., CHRISTIE B. (éds) – Applying cognitive psychology to user-interface design. New-York, Wiley et sons, 1987, pp. 13-54.
- HELANDER M.G., RUPP B.A. – An overview of standards and guidelines for visual display terminal. *Applied Ergonomics,* 1984, 3, 15, pp. 185-195.
- HOC J.M. – L'étude psychologique de l'activité de programmation : une revue de question. *Techniques et Sciences Informatiques,* 1982, pp. 383-392.
- HOC J.M. – Aides logicielles à la résolution de problème et assistance aux activités de la planification. Colloque Ergonomie en informatique, Nivelles, 25-27 novembre 1985, 16 p.

- Hutchins E., Hollan J., Norman D. – Direct manipulation interfaces. In : Norman D.A., Draper S.W. (éds) – User centered system design, new perspectives on human-computer interaction. Londres, Lawrence Erlbaum Associates Publishers, 1986, pp. 87-124.
- Janet E. – Ergonomie des dispositifs d'entrée d'informations par repérage sur écran cathodique. *Le Travail Humain,* 1982, 45, 2, pp. 285-305.
- Kerguelen A. – Aménagement ergonomique d'un logiciel en cours de conception, critères et méthodologie de conception des systèmes informatiques. Paris, Laboratoire de psychologie du travail de l'EPHE, 1983, 41 p.
- Le Bourgeois A., Valentin A. – Vers un nouveau dialogue avec l'ordinateur. *Lettre d'Information de l'Agence Nationale pour l'Amélioration des Conditions de Travail,* octobre 1986, 112, pp. 2-9.
- Landauer T. – Relations between cognitive psychology and computer system design. In : Carroll J. (éd.) – Interfacing thought : cognitive aspect of human computer interaction. Cambridge, The MIT Press, 1987, pp. 1-25.
- Lee E., MacGregor J. – Minimizing user search time in menu retrieval systems. *Human Factors,* 1985, 27, 2, pp. 157-162.
- Levis C., Norman D.A – Designing for error. In : Norman D.A., Draper S.W. (éds) – User centered system design, new perspectives on human-computer interaction. Londres, Lawrence Erlbaum Associates Publishers, 1986, pp. 411-431.
- Mayer R. – Cognitive aspects of learning and using a programming language. In : Carroll J. (éd.) – Interfacing thought : cognitive aspect of human computer interaction. Cambridge, The MIT Press, 1987, pp. 61-79.
- Mazoyer B., Salembier P. – Les principales techniques de recueil de données en ergonomie des logiciels. In : Alzera C., Christol J., Falzon P., Mazoyer B., Pinsky L., Salembier P. (éds) – Ergonomie des logiciels : un atout pour la conception des systèmes informatiques. Paris, La documentation française, collection Les cahiers « Technologie, emploi, travail », 1987a, pp. 22-26.
- Mazoyer B., Salembier P. – La maniabilité : une dimension mesurable de la qualité des logiciels. In : Alzera C., Christol J., Falzon P., Mazoyer B., Pinsky L., Salembier P. (éds) – Ergonomie des logiciels : un atout pour la conception des systèmes informatiques. Paris, La documentation française, collection Les cahiers « Technologie, emploi, travail », 1987b, pp. 47-52.
- Michard A. – Les logiciels d'explication automatique. *Le Monde Informatique,* 25 novembre 1985, pp. 35-36.
- Michard A., Giboin A., Mais C., Verdret P. – PLANEX : système d'aide à la planification. Sophia Antipolis, INRIA, rapport technique nº 95, 1988, 44 p.
- Miyata Y., Norman D.A. – Psychological issues in support of multiple activities. In : Norman D.A., Draper S.W. (éds) – User centered system design, new perspectives on human-computer interaction. Londres, Lawrence Erlbaum Associates Publishers, 1986, pp. 265-283.
- Morel P. – La formation pour une bureautique efficace. Metz, Laboratoire de recherche en sciences humaines et sociales, rapport interne, 1987, 74 p.
- Nakaseko M., Grandjean E., Hunting W., Gierer R. – Studies on ergonomically designed alphanumeric keyboards. *Human Factors,* 1985, 27, 2, pp. 175-187.
- Nedzynski S. – International trade union guidelines on visual display units. Genève, Congrès, 29-30 octobre 1984, 91 p.
- Norman D.A. – Cognitive engineering. In : Norman D.A., Draper S.W. (éds) – User centered system design, new perspectives on human-computer interaction. Londres, Lawrence Erlbaum Associates Publishers, 1986, pp. 31-61.
- Norman D.A., Draper S.W. – User centered system design, new perspectives on human-computer interaction. Londres, Lawrence Erlbaum Associates Publishers, 1986, 525 p.
- Pastre O. – L'informatisation et l'emploi. Paris, La Découverte, 1984, 127 p.
- Patesson R., Karnas G., Springael B. – Etude de l'activité et des représentations du travail à l'occasion des saisies de masse. Communication au XXᵉ congrès de la SELF, Genève, 1984, 12 p.
- Phan Huy Dong - Conception de logiciel : question de cohérence, question de démarche, question d'outil (1). *L'Informatique Professionnelle,* 1985a, 38, pp. 69-94.
- Phan Huy Dong - Conception de logiciel : question de cohérence, question de démarche, question d'outil (2). *L'Informatique Professionnelle,* 1985b, 39, pp. 47-61.
- Piganiol C. – L'ergonomie des logiciels (1). *L'Informatique Professionnelle,* 1984a, 22, pp. 103-114.
- Piganiol C. – L'ergonomie des logiciels (2). *L'Informatique Professionnelle,* 1984b, 23, pp. 47-71.
- Pynte J. – Pour une approche psycholinguistique du « dialogue homme-machine. *Cahier de Psychologie Cognitive,* 1984, 4, 2, pp. 127-149.
- Rialle V. – Ergonomie cognitive et base de données. Communication au Forum IA'88, Paris, 11-13 octobre 1988, 12 p.
- Richard J.F. – Logique de fonctionnement, logique d'utilisation. Le Chesnay, INRIA, 1983, rapport nº 202, 47 p.
- Robert J.M. – Some highlight of learning by exploration. International scientific conference : Work with display units, Stockholm, 12-15 mai 1986, pp. 348-353.
- Salembier P., Claes G. – Apports de l'ergonomie à la conception d'un système EIAO. In : Alzera C., Christol J., Falzon P., Mazoyer B., Pinsky L., Salembier P. (éds) – Ergonomie des logiciels : un atout pour la conception des systèmes informatiques. Paris, La documentation française, collection Les cahiers « Technologie, emploi, travail », 1987, pp. 15-21.
- Scapin D.L. – Human factors aspects in the design of command languages. In : Naffah N. (éd.) – Office information systems. North-Holland/INRIA, 1982, pp. 541-549.
- Scapin D.L. – Ergonomie du logiciel : situation et thèmes d'études. Le Chesnay, INRIA, document interne, 1984, 7 p.
- Scapin D.L. – Guide ergonomique de conception des interfaces homme-ordinateur. Le Chesnay, INRIA, 1985, 64 p.
- Senach B. – Une approche ergonomique de l'informatique. *Le Journal des Psychologues,* 1986, 41, pp. 36-40.
- Senach B. – Intelligence des logiciels d'aide à l'utilisation et modélisation de l'activité des utilisateurs. In : Alzera C., Christol J., Falzon P., Mazoyer B., Pinsky L., Salembier P. (éds) – Ergonomie des logiciels : un atout pour la conception des systèmes informatiques. Paris, La documentation française, collection Les cahiers « Technologie, emploi, travail », 1987, pp. 77-89.
- Shackel B. – Dialogues and language ; can computer ergonomics help ? *Ergonomics,* 1980, 23, 9, pp. 857-880.
- Shackel B. – Ergonomics in information technology in Europe ; a review. *Behaviour and Information Technology,* 1985, 4, 4, pp. 263-287.
- Sperandio J.C. – L'ergonomie du travail mental. Paris, Masson, 1984, 130 p.
- Sperandio J.C. – L'ergonomie du travail informatisé. In : Traité de psychologie du travail. Paris, PUF, 1987a, pp. 161-176.
- Sperandio J.C. – Introduction à l'ergonomie des logiciels. In : Alzera C., Christol J., Falzon P., Mazoyer B., Pinsky L., Salembier P. (éds) – Ergonomie des logiciels : un atout pour la conception des systèmes

informatiques. Paris, La documentation française, collection Les cahiers « Technologie, emploi, travail », 1987b, pp. 15-21.

- STREITZ N.A. – Cognitive compatibility as central isssue in human-computer interaction : theoretical framework and empirical finding. In : SALVENDY G. (éd.) – Cognitive engineering in the design of human computer interaction and expert system. Amsterdam, Elsevier science publishers, 1987, pp. 75-82.

- THEUREAU J., PINSKY L. – Paradoxe de l'ergonomie de conception et logiciel informatique. *Revue des Conditions de Travail,* 1984, 9, pp. 25-31.

- TURNER J.A., KARASEK Jr R.A. – Software ergonomics : effects of computer application design parameters on operator task performance and health. *Ergonomics,* 1984, 27, 6, pp. 663-690.

- VALENTIN A., LUCONGSANG R. – L'ergonomie des logiciels. Paris, Agence nationale pour l'amélioration des conditions de travail (ANACT), 1987, 118 p.

- VANNESTE C. – Maquettage et prototypage pour la conception d'interfaces homme-ordinateur. In : ALZERA C., CHRISTOL J., FALZON P., MAZOYER B., PINSKY L., SALEMBIER P. (éds) – Ergonomie des logiciels : un atout pour la conception des systèmes informatiques. Paris, La documentation française, collection Les cahiers « Technologie, emploi, travail », 1987, pp. 53-56.

- WILLIGES R.C., EHRICH R.W. – Human computer dialogue design. In : SALVENDY G. (éd.) – Advances in human factor / Ergonomics 2. New-York, Elsevier, 384 p.

- WINOGRAD T. – Les logiciels de traitement des langues naturelles. *Pour la Science,* 1984, 85, pp. 90-103.

- WISNER A. – Préface. In : ALZERA C., CHRISTOL J., FALZON P., MAZOYER B., PINSKY L., SALEMBIER P. (éds) – Ergonomie des logiciels : un atout pour la conception des systèmes informatiques. Paris, La documentation française, collection Les cahiers « Technologie, emploi, travail », 1987, pp. 7-9.

Reçu en avril 1988, accepté après révision en octobre 1989 ■

INSTITUT NATIONAL DE RECHERCHE ET DE SÉCURITÉ
30, rue Olivier-Noyer, 75680 Paris cedex 14
Tiré à part des Cahiers de notes documentaires, 2e trimestre 1990, no 139 - ND 1780 - No CPPAP 804 AD/PC/DC du 14-03-85
Directeur de la publication : D. MOYEN
ISSN 0007-9952 - ISBN 2-85599-935-9